iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** VII **Month of publication:** July 2023

DOI: https://doi.org/10.22214/ijraset.2023.54916

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VII Jul 2023- Available at www.ijraset.com

Personalized Font Generation using Deep Learning Neural Networks

Chirag Desai[1], Darsh Pandya[2], Ansh Khatri[3], Narayan Prajapati[4]
Bachelor of Technology in Information Technology, K.J. Somaiya College of Engineering, Mumbai, Maharashtra

Abstract: Personalized font generation is an emerging technology that aims to create unique and customized fonts based on an individual's preferences and characteristics. This process involves designing a font from scratch, including selecting the appropriate style, weight, and size. Personalized features such as letter shapes, ligatures, and kerning are then implemented to create a font that is unique to the individual. Personalized font generation has the potential to revolutionize the way we think about typography. By creating fonts that are tailored to an individual's preferences, we can enhance the user experience and create more engaging and personalized content. This technology is particularly relevant in today's digital age, where the use of typography is increasingly important in everything from social media posts to marketing materials. In summary, personalized font generation is a promising new technology that offers a unique and tailored typography experience. By combining traditional font design techniques with advanced technologies such as machine learning and artificial intelligence, we can create fonts that are truly one-of-a-kind and help to elevate our communication efforts.
Keywords: font, ligatures, kerning, machine learning, communication

I. INTRODUCTION

Our objective is to generate personal Font of the User's Handwriting, with the help of neural network. User input of lowercase and uppercase alphabets, special characters and digits will be used to generate user's handwriting. Typing is a lot quicker than writing and hence it will save time and efforts of writers, students etc. Users with diseases like Parkinson's and partially disabled people can write in their own handwriting. It will help for voice to text conversion with personalized handwriting in future.

Dysgraphia is a disability that affects both children and adults' fine motor abilities. Our project looked into how neural networks could be used to help persons with this disorder smooth up their handwriting so they could communicate more effectively. This project will result in an application that takes handwriting data in the form of sequential stroke data and generates a collection of stroke data that represents new smooth handwriting. This application will not only benefit persons with Dysgraphia, but it may also spur further research in a variety of other handwriting and drawing applications.

Our application – WeWrite, is a personalized font generator. The font generator must generate a personal font based on the handwriting of the person. The font generator must take handwriting inputs in the form of letters (a-z & A-Z) and digits (0-9). The system must store the personal font in order for the user to convert normal font into the personalized one. User can create an account and download their personalized font.

In all facets of daily life, from education and communication to art and personal identity, handwriting, a fundamental form of human expression, has both utilitarian and aesthetic significance. There is a lot of interest in the domains of artificial intelligence and machine learning on the creation of human-like handwriting using neural networks. Numerous applications, including automatic document generation, the creation of custom fonts, and interactive human-computer interfaces, could be completely transformed by handwriting generation. The fascinating field of handwriting generation using neural networks is explored in this introduction, along with its importance, difficulties, and prospective effects on various disciplines.

The practice of handwriting has developed over the years, giving rise to a variety of styles and distinctive variances. However, due to the dynamic and complicated structure of the strokes, the wide variety of writing styles, and the need to preserve readability and coherence, creating human-like handwriting digitally is a challenging endeavor. Traditional rule-based or statistical methods for creating handwriting have had trouble replicating the subtleties and naturalness of actual handwriting.

Recurrent neural networks (RNNs) and their derivatives, which have recently emerged as potent deep learning approaches, have provided intriguing solutions to this problem. RNNs have shown success in a variety of natural language processing applications and can model sequential data. By extending these capabilities to handwriting generation, researchers aim to develop models that can mimic the artistic nuances and temporal dependencies present in human handwriting.

The objective of handwriting generation using neural networks is twofold: (1) to produce visually convincing and contextually appropriate handwriting styles, and (2) to maintain legibility and coherence in generated text. Achieving this balance between creative freedom and adherence to legibility rules presents a compelling research problem.

This paper delves into the state-of-the-art techniques employed in handwriting generation. We discuss the architectural components of recurrent neural networks, such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), which enable the capture of long-range dependencies and context awareness necessary for generating fluid and coherent handwriting sequences. Additionally, we explore the integration of attention mechanisms, allowing the model to focus on relevant parts of the input sequence during generation, further enhancing the naturalness of the output.

Furthermore, we investigate the importance of building comprehensive datasets of diverse handwriting samples, their preprocessing, and augmentation to train robust handwriting generation models. We also explore how fine-tuning and transfer learning techniques can be employed to adapt the model to specific handwriting styles or individual writing characteristics.

The potential applications of handwriting generation using neural networks are far-reaching. From aiding in the automation of document generation to creating personalized handwriting for digital communication and artistic endeavors, this technology has the potential to revolutionize the way we interact with written text.

In conclusion, this research area holds great promise in pushing the boundaries of what neural networks can accomplish in terms of human-like creativity and expression. The exploration of handwriting generation using neural networks not only furthers our understanding of artificial intelligence but also paves the way for practical applications that can positively impact various domains. As researchers continue to innovate and refine these techniques, we anticipate significant advancements in the field of handwriting generation, shaping the future of human-computer interaction and visual communication.

II. LITERATURE SURVEY

Lakshmi M. Gadhikar et. al. gave us an approach to tackle this problem and helped us segment the project into sections and the order in which those sections will be implemented. It also introduced us to “Moore-Neighbour Tracing Algorithm”. [2]

Prathiba Sharma et. al. told the different methods used for detecting edges in image processing, it told how methods like Sobel, Prewitt (old methods) are sensitive to noise and the “Canny Algorithm” (which is still widely used and competes with Moore-Neighbour Tracing Algorithm) takes more computational time. [7]

Hideaki Hayashi et. al. proposed GlyphGAN: style-consistent font generation based on generative adversarial networks (GANs). GANs are a framework for learning a generative model using a system of two neural networks competing with each other. One network generates synthetic images from random input vectors, and the other discriminates between synthetic and real images. The motivation of this study is to create new fonts using the GAN framework while maintaining style consistency over all characters. In GlyphGAN, the input vector for the generator network consists of two vectors: character class vector and style vector. The former is a one-hot vector and is associated with the character class of each sample image during training. The latter is a uniform random vector without super-vised information. In this way, GlyphGAN can generate an infinite variety of fonts with the character and style independently controlled. Experimental results showed that fonts generated by GlyphGAN have style consistency and diversity different from the training images without losing their legibility. Deals with generating new fonts which are not based on inputs given by us in the form of handwriting, but by using system of 2 neural networks where the input vector has 2 parts out of which the first consists of trained samples and second will have random style (which we are not looking for). Nevertheless, this paper gave us an insight as to have neural networks can be incorporated in a project related to us. [5]

Alex Greaves et. al. shows how Long Short-term Memory recurrent neural networks can be used to generate complex sequences with long-range structure, simply by predicting one data point at a time. The approach is demonstrated for text and online handwriting. It is then extended to handwriting synthesis by allowing the network to condition its predictions on a text sequence. The resulting system is able to generate highly realistic cursive handwriting in a wide variety of styles. [6]

Takeo Igarashi et. al. This is an old paper which employs a technique which extracts the skeleton of the alphabets once we give it individual alphabets in the form of a template, then we change make changes to the skeleton like varying the length and the breadth. Although this paper is good for non- cursive handwriting, it does not work for cursive handwriting. [1]

Fogel, Sharon, et. al. shows a variation of GAN again known as ScrabbleGAN, which is a combination of GAN and CRNN (Convolutional Recurrent Network). Although this paper an generate sequences of cursive handwriting and we can make some changes in the form of length of the alphabets but it doesn’t allow users to input their own handwriting styles which will reflect in the handwriting. [3]

Kundu, Soumyadeep, et. al. wrote a paper on a variant of the GAN architecture, we thus tried to incorporate GAN into our project, this was before we came to the conclusion that RNN is the best way to move forward and generate sequences. [4] A. Kumar, M. Madaan, et. al. wrote a paper implementing the original Alex Greaves paper. This paper was necessary as the original paper is more than 5 years old and although it is still widely used, a current implementation helped us understand the modules we must code, it also made us know that the old architecture of Alex Greaves is still valid and can be used effectively.[8]

TABLE I. LITERATURE OVERVIEW

Sr. No.	Author	Year	Methodology
1	Suveeranont, Rapee, and Takeo Igarash	2010	Employs a technique which captures the skeleton of an alphabet
2	Lakshmi M. Gadhikar, Ajinkya Shukla,	2013	Introduced us to Moore-Neighbor Tracing Algorithm
3	Sharon Fogel, Hadar Averbuch-Elor, Sarel Cohen, Shai Mazor, Roee Litman	2013	Introduced us to ScrabbleGAN, which is variation of GAN
4	Kundu, Soumyadeep	2020	It again incorporates GAN architecture but done on Chinese character
5	Hayashi, Hideaki, Kohtaro Abe, and Seiichi Uchida	2019	It again suggests a variation of GAN called GlyphGAN, it also takes a style vector as input.
6	Graves, Alex	2013	It uses LSTM to generate realistic sequences based on the dataset and user input.
7	Sharma, Pratibha, Manoj Diwakar, and Niranjan Lal.	2013	It introduced us to methods like Sobel, Prewitt and the Canny Algorithm
8	A. Kumar, M. Madaan, S. Kumar, A. Saha and K.Gupta	2022	It is a re-implementation of the Alex Graves paper with mi-nor modifications in the synthesis model.

III. METHODOLGY

A. Data Collection and Preprocessing

The choice of the IAM (Iam Handwriting) dataset for personalized font generation is driven by its rich and diverse collection of handwritten samples, making it ideal for capturing individual writing styles. The IAM dataset contains a wide range of handwriting samples from different writers, encompassing various writing contexts and languages. This diversity allows the neural network to learn and generalize across a vast spectrum of handwriting variations, ensuring the model's ability to produce personalized fonts for individuals with distinct writing characteristics. Additionally, the availability of ground truth labels in the IAM dataset facilitates supervised training, enabling the model to learn from high-quality, annotated data. By leveraging the IAM dataset, the personalized font generation model can capture the unique subtleties and artistic nuances present in each individual's handwriting, empowering users to create personalized and authentic fonts that resonate with their own identity and expression.

B. Neural Network Architecture Selection:

We have chosen The LSTM model which is a special kind of Recurrent Neural Network (RNN) to generate the font. The LSTM (Long Short-Term Memory) model of recurrent neural networks is an excellent choice for personalized font generation due to its inherent ability to capture long-range dependencies and temporal patterns in sequential data, such as handwriting strokes. In the context of personalized font generation, individuals exhibit unique writing styles and penmanship nuances that span across multiple characters and strokes. LSTM's memory cell architecture allows it to retain and utilize information from distant past time steps, making it well-suited for capturing the coherence and fluidity of an individual's handwriting across the entire text. By leveraging LSTM, the personalized font generation model can better grasp the subtle variations and intricacies of each individual's writing style, resulting in more accurate and realistic personalized fonts. Additionally, LSTM's flexibility allows for efficient training on datasets of varying lengths, accommodating different writing samples and ensuring adaptability to the diverse writing styles encountered in real-world personalized font generation scenarios. Below is fig.1 that shows the architecture of our chose model.

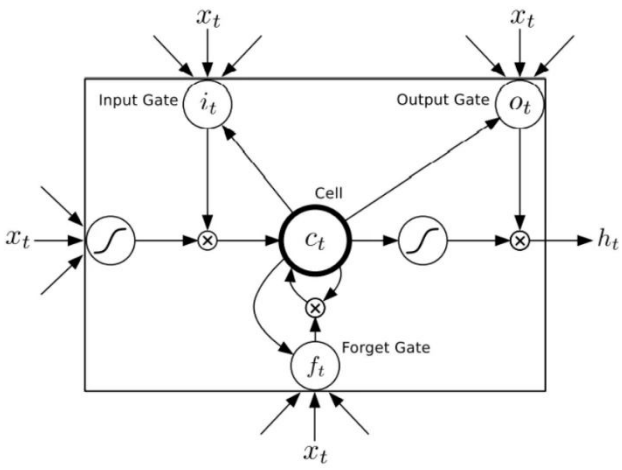


Fig. 1. Architecture of LSTM model

C. Training the Neural Network

After choosing the architecture of the model which are we going to use for our font generation, we have to train the model accurately to achieve an accurate output.

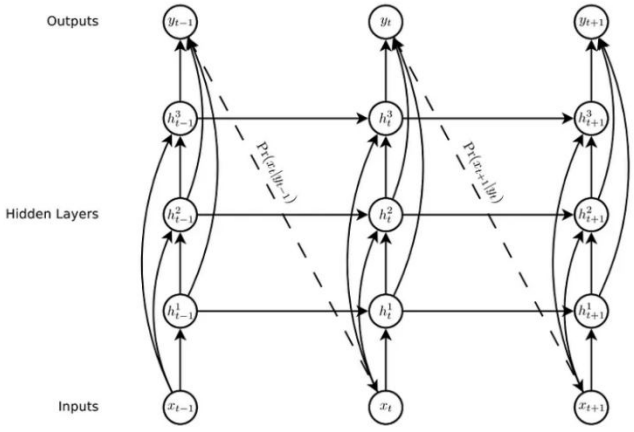


Fig. 2. General architecture of our proposed model

we can see the RNNs architecture, where x (x1, x2, ... xT) input cells are also directly passed to the 2nd and 3rd hidden layers. As we know, RNN takes the previous output as input in the next cell which can be seen in the above figure (dashed line) in the form of probability (probability of next input given the previous output). The equation for the hidden layers will be,

$$h_t^1 = \mathcal{H}\left(W_{ih^1}x_t + W_{h^1h^1}h_{t-1}^1 + b_h^1\right)$$

Fig. 3.Equation for the hidden layers

Where the H represents the hidden layer activation function, W represents the weight of the connections (lines) (e.g. W(i, h1): weight matrix of the line connecting the input to the 1st hidden layer), Xt is the Tth term of the input vector, and b denotes the bias vector. The output vector y is used to determine the predictive distribution (Pr(X t‖Y t − 1))f or the next input.

$$\hat{y}_t = b_y + \sum_{n=1}^{N} W_{h^n y} h_t^n \qquad \mathcal{L}(\mathbf{x}) = -\sum_{t=1}^{T} \log \Pr(x_{t+1}|y_t)$$
$$y_t = \mathcal{Y}(\hat{y}_t)$$

Fig. 4.Formula for output vector

Above we can see the loss (L(x)) and the output vector (Y t) equation where, gamma is the output layer function. In the next section we will see the equation of the predictive distribution(Pr(Xt‖Yt−1)).

D. Hyperparameter Tuning:

Hyperparameter tuning for recurrent neural networks (RNNs) in personalized font generation involves optimizing several key hyperparameters to achieve the best performance and generalization of the model. Some of the essential hyperparameters to tune are as follows:

Learning Rate: The learning rate determines the step size taken during optimization to update the model's parameters. Tuning the learning rate involves experimenting with different values (e.g., 0.1, 0.01, 0.001) to find the optimal balance between faster convergence and avoiding overshooting the optimal solution. Set to 0.1.

Number of Hidden Units: The number of hidden units in the LSTM layers impacts the model's capacity to learn complex patterns in the data. Tuning this hyperparameter involves trying various values (e.g., 64, 128, 256) to strike a balance between model complexity and overfitting. Set to 256.

Number of LSTM Layers: The depth of the LSTM architecture influences the network's ability to capture long-term dependencies. Experimenting with the number of LSTM layers (e.g., 1, 2, 3) helps find the optimal depth for the personalized font generation task. Set to 3

Batch Size: The batch size determines the number of samples processed together during each training iteration. Tuning this hyperparameter involves exploring different batch sizes (e.g., 16, 32, 64) to identify the trade-off between computational efficiency and convergence stability. Set to 32.

Dropout Rate: Dropout is a regularization technique used to prevent overfitting by randomly setting a fraction of LSTM units to zero during training. Tuning the dropout rate (e.g., 0.2, 0.5, 0.7) helps strike a balance between reducing overfitting and maintaining model expressiveness. Set to zero.

Sequence Length: The sequence length determines the number of characters or handwriting strokes fed into the model during each forward and backward pass. It is essential to experiment with different sequence lengths (e.g., 50, 100, 150) to find an optimal trade-off between memory requirements and capturing long-range dependencies. Set to 100.

Optimization Algorithm: The choice of optimization algorithm, such as Adam, RMSprop, or SGD, can significantly impact training efficiency and convergence. Tuning the optimization algorithm involves experimenting with various options to identify the most suitable one for personalized font generation. We have used SGD

Number of Epochs: The number of training epochs determines how many times the model iterates through the entire dataset during training. It is crucial to find an appropriate number of epochs to prevent underfitting or overfitting. Found an Optimal count at 120 epochs. The hyperparameter tuning process typically involves using techniques like grid search, random search, or Bayesian optimization to explore a wide range of hyperparameter combinations and identify the configuration that yields the best performance and generalization on a validation set. Proper hyperparameter tuning is essential to ensure the personalized font generation model achieves its highest potential in accurately capturing and replicating individual writing styles.

E. Deployment and Application

Once the handwriting synthesis model has been trained thoroughly and is achieving satisfactory performance, it can be deployed in various applications including document generation, artistic expression, interactive interfaces, and personalized communication platforms. After Deployment the user enters his/her handwriting input, or shares a clear handwritten image. With that the user has to label what the picture or handwriting indicates and set a value to the handwriting. Once the input data is received from the user the handwriting data is used on the handwriting synthesis model and the model tries to mimic the handwriting input of the user accurately. The user gets to decide if he/she wants to use their font or our default font to generate the handwriting. The model will generate 5 outputs generated based on the 'Bias' input (a value from 0-10). The Bias is used to get some variance in order to get the 5 different outputs. The 5 different outputs are then displayed to the user and the user is allowed to pick or choose whichever output font is the most similar to his original handwriting

IV. OUTPUT

Our neural network model used to generate user handwriting was used to create user outputs. We ran multiple test cases to check how well our model was working. Our model displayed 5 variant outputs based on the values of user inputs. The output landing page is shown below with a few of our respective test cases.

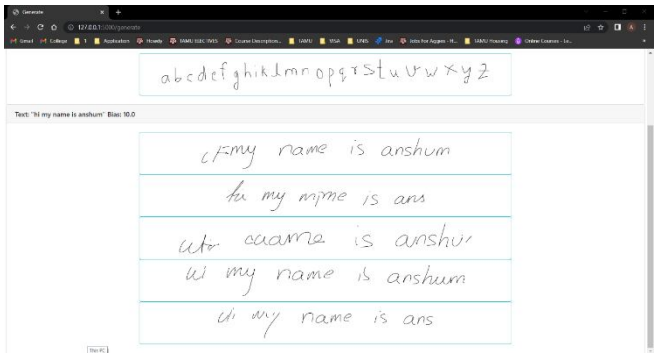


Fig. 5.Output of 'hi my name is anshum" based on "abc…yz"

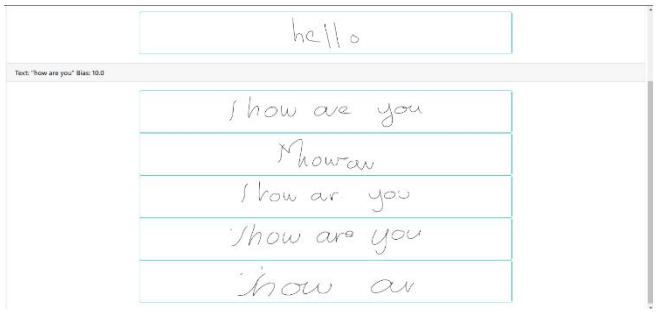


Fig. 6.Output of "how are you" based on "hello"

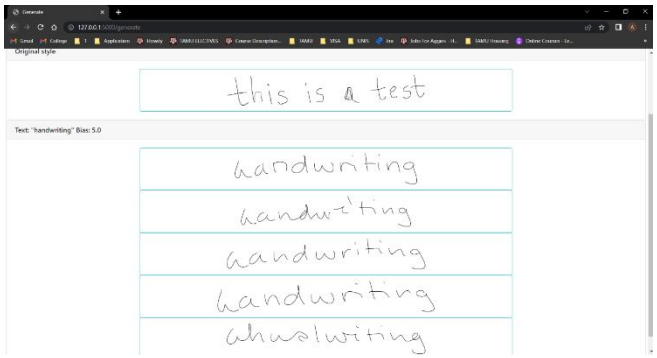


Fig. 7.Output of "handwriting" based on "this is a test"

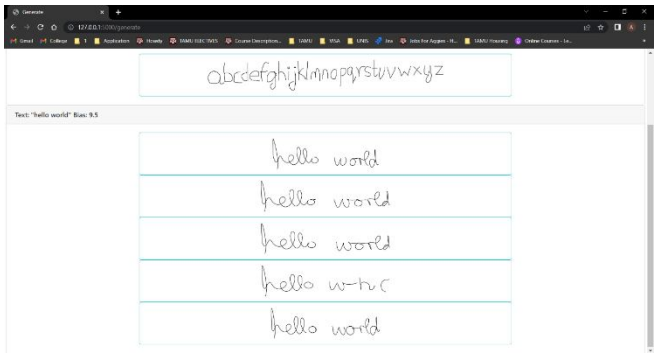


Fig. 8.Output of "hello world" based on "abc…yz"

Fig. 5,6,7,8, show us the outputs of our trained model.

V. CONCLUSION

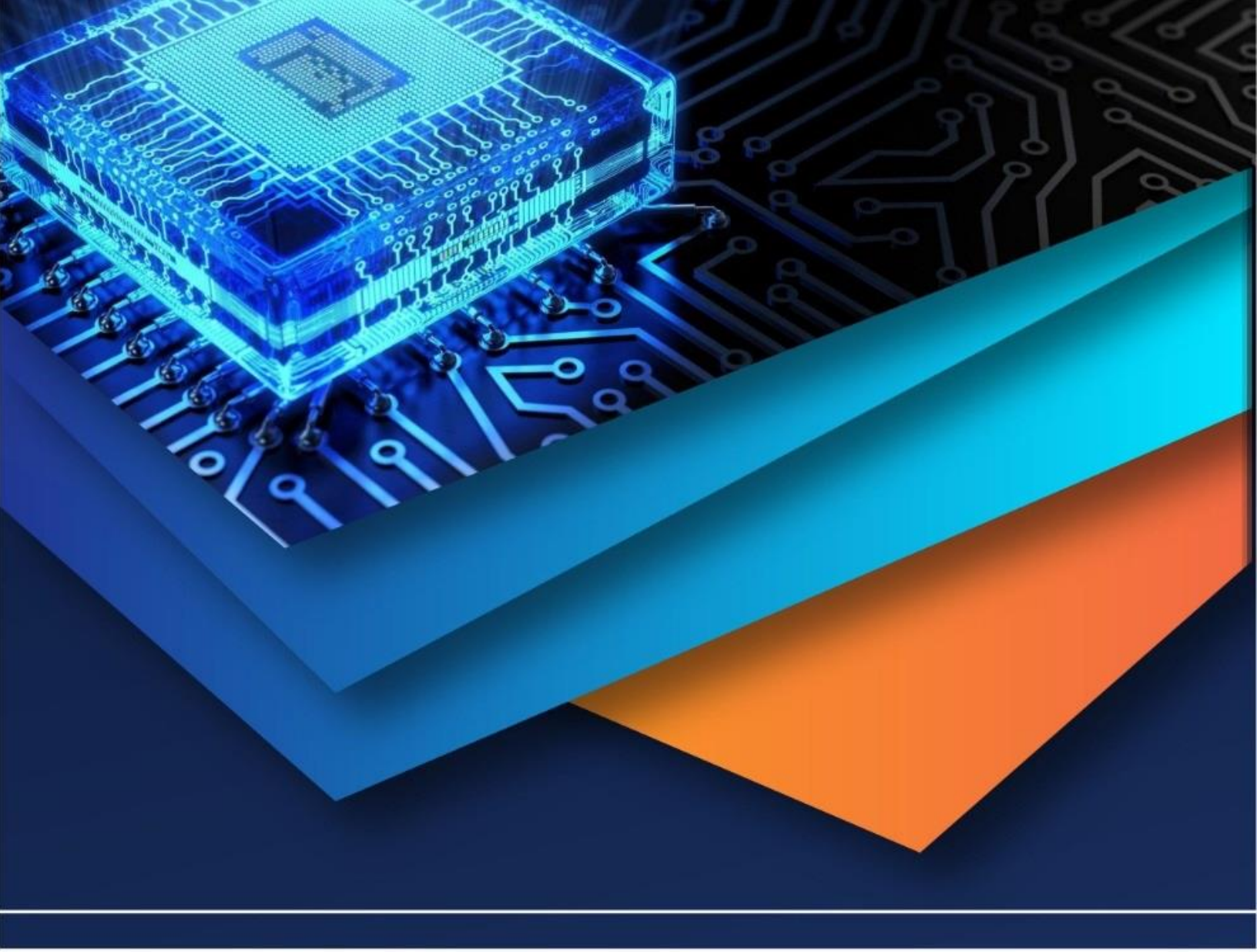
Artificial intelligence is a rapidly evolving discipline that is seeing a slew of new developments. The field of handwriting generation has made significant development in recent years owing to the application of advanced approaches such as RNNs, LSTMs, RL, and GANs. We discovered that the LSTM model gives an average representation of the user handwriting. It mimics user handwriting with the help of user input data. The handwriting synthesis model that we created has a mean squared error of 44%. Dysgraphia is a disability that affects both children and adults' fine motor abilities. Our project looked into how neural networks could be used to help persons with this disorder smooth up their handwriting so they could communicate more effectively. This project will result in an application that takes handwriting data in the form of sequential stroke data and generates a collection of stroke data that represents new smooth handwriting. This application will not only benefit persons with Dysgraphia, but it may also spur further research in a variety of other handwriting and drawing applications.

VI. FUTURE SCOPE

Personalized Font Generation currently has a massive potential for future work. We encountered a lot of ideas and have various ways in which the algorithm for font generation can be made better as per user needs, to provide a more accurate font of the user. Work can be done in variety of areas, such as preparation of a better dataset, and better preprocessing of the data, to make the ligatures more prominent. Whole set of characters can be taken as an input from the user, so that there's much clarity for the system to understand user's styles and handwriting. Further, user can be allowed to download the output of their own handwriting for future use, if any in the form .ttf files. And if users are allowed to download their personal files, we also need to have security in the form of personal accounts, so that the users font is not misused for malpractices.

REFERENCES

[1] Suveeranont, Rapee, and Takeo Igarashi. "Example-based automatic font generation." Smart Graphics: 10th International Symposium on Smart Graphics, Banff, Canada, June 24-26, 2010 Proceedings 10. Springer Berlin Heidelberg, 2010
[2] Lakshmi M. Gadhikar, Ajinkya Shukla, 2013, Implementation of Automatic Font Generation, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH TECHNOLOGY (IJERT) Volume 02, Issue 10 (October 2013).
[3] Fogel, Sharon, et al. "Scrabblegan: Semi-supervised varying length handwritten text generation." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020.
[4] Kundu, Soumyadeep, et al. "Text-line extraction from handwritten document images using GAN." Expert Systems with Applications 140 (2020): 112916.
[5] Hayashi, Hideaki, Kohtaro Abe, and Seiichi Uchida. "GlyphGAN: Style-consistent font generation based on generative adversarial networks." Knowledge-Based Systems 186 (2019): 104927.
[6] Graves, Alex. "Generating sequences with recurrent neural networks." arXiv preprint arXiv:1308.0850 (2013).
[7] Sharma, Pratibha, Manoj Diwakar, and Niranjan Lal. "Edge detection using Moore neighborhood." International Journal of Computer Applications 61.3 (2013).
[8] A. Kumar, M. Madaan, S. Kumar, A. Saha and K. Gupta, "Handwriting Generation and Synthesis using Recurrent Neural Networks," 2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2022, pp. 1-8, doi: 10.1109/ICCCNT54827.2022.998445



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY